Projeto Captura de Pacotes

1 Endereço IP da aplicação no computador e endereço IP da máquina remota.

Computador: 172.31.222.216

Máquina remota: 212.227.179.107

	http								
No).	Time	Source	Destination	Protocol	Length Info			
+	6445	54.493139527	212.227.170.107	172.31.222.216	HTTP	1120 HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)			
+	6457	54.504608653	172.31.222.216	212.227.170.107	HTTP	402 GET /flieger.gif%22 HTTP/1.1			
1	6626	54.982757446	212.227.170.107	172.31.222.216	HTTP	1440 HTTP/1.1 200 OK (JPEG JFIF image)			
1	6653	55.052117453	212.227.170.107	172.31.222.216	HTTP	1440 HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)			
1	6661	55.086288349	212.227.170.107	172.31.222.216	HTTP	1050 HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)			
1	6671	55.097755170	212.227.170.107	172.31.222.216	HTTP	809 HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)			
- 1	6769	55.319943271	212.227.170.107	172.31.222.216	HTTP	359 HTTP/1.1 200 OK (JPEG JFIF image)			
1	6807	55.390998466	212.227.170.107	172.31.222.216	HTTP	546 HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)			
-	6815	55.393728492	172.31.222.216	212.227.170.107	HTTP	398 GET /relief.jpg HTTP/1.1			

Figura 1: O campo *Source* do pacote destacado representa o IP da máquina remota e o campo *Destination*, o IP da aplicação no computador. OK é enviado pelo servidor(máquina remota), GET é enviado pelo cliente(aplicação no computador).

2 Três protocolos que foram observados nos pacotes capturados e as funções dos mesmos.

	Apply a display filter <ctri-></ctri->								
No	. Time	Source	Destination	Protocol	Length Info				
1	7038 56.516942065	212.227.170.107	172.31.222.216	TCP	1440 [TCP segment of a reassembled PDU]				
	7039 56.516977195	172.31.222.216	212.227.170.107	TCP	66 41874 → 80 [ACK] Seq=341 Ack=56793 Win=150784 Len=0 TSval=520112 TSecr=589475147				
	7040 56.538509542	212.227.170.107	172.31.222.216	TCP	1440 [TCP segment of a reassembled PDU]				
1.	7041 56.538550539	172.31.222.216	212.227.170.107	TCP	66 41874 → 80 [ACK] Seg=341 Ack=58167 Win=153600 Len=0 TSval=520117 TSecr=589475148				
1.	7042 56.558090454	212.227.170.107	172.31.222.216	TCP	1440 [TCP segment of a reassembled PDU]				
1.5	7043 56.558135851	172.31.222.216	212.227.170.107	TCP	66 41874 → 80 [ACK] Seq=341 Ack=59541 Win=156544 Len=0 TSval=520122 TSecr=589475148				
1	7044 56.628013382	172.31.192.13	172.31.192.127	UDP	62 2008 → 2008 Len=20				
1	7045 56.654766575	212.227.170.107	172.31.222.216	TCP	1440 [TCP segment of a reassembled PDU]				
1.5	7046 56.654805895	172.31.222.216	212.227.170.107	TCP	66 41874 → 80 [ACK] Seq=341 Ack=60915 Win=159488 Len=0 TSval=520146 TSecr=589475148				
	7047 56.658353833	212.227.170.107	172.31.222.216	HTTP	695 HTTP/1.1 200 OK (GIF89a) (GIF89a) (image/gif)				
1	7048 56.658374925	172.31.222.216	212.227.170.107	TCP	66 41874 → 80 [ACK] Šeq=341´Ačk=61544´Win=162176´Len=0 TSval=520147 TSecr=589475148				

Figura 2: Na figura acima, podem se perceber os tipos de protocolo TCP, UDP E HTTP.

Protocolo TCP (Protocolo de Controle de Transmissão), protocolo implementado na camada de transporte a qual é responsável por fornecer comunicação logica entre dois hospedeiros, o TCP faz isso de forma orientada à conexão, sendo um protocolo que garante uma entrega confinável dos pacotes, alem disto o TCP implementa um controle de congestionamento um uso melhor e mais justo da rede.

Protocolo UDP (Protocolo de Datagrama de Usuário), também um protocolo da camada de transporte, porém o UDP implementa as responsabilidades desta camada de uma forma diferente, o UDP não é orientado a conexão e muito menos tem um controle de congestionamento, não garantindo uma entrega confiável, mas em compensação é um protocolo muito mais simples e com um *overhead* menor.

Protocolo HTTP (Protocolo de Transferência de Hipertexto),o protocolo da camada de aplicação, responsável pela troca de mensagens entre dois sistemas finais, mais comumente utilizado em paginas web.

3 Esquema de encapsulamento até a camada de enlace de uma sequência de pacotes capturados.

Vemos nas figuras a seguir o encapsulamento de três pacotes diferentes, o primeiro pacote usa um protocolo da camada de aplicação o HTTP, o TCP na camada de transporte, o IP na camada de rede, o Ethernet II na camada de enlace e tudo isso é transmitido em uma camada física. Já no segundo pacote é usado o UDP na camada de transporte, e por fim o ultimo pacote usa TCP na camada de transporte e não possui camada de aplicação. É interessante notar que por mais que em uma camada o protocolo utilizado seja diferente as outras camadas continuam oferecendo o mesmo serviço, mostrando a importância e a modularidade de uma arquitetura em camadas.

```
Wireshark-Packet7047-pacotes_projeto1

Frame 7647: 695 bytes on wire (5560 bits), 695 bytes captured (5560 bits) on interface 0

Ethernet II, Src: ArubaNet_02:38:10 (00:1a:1e:02:38:10), Dst: HonHalPr_94:8f:e5 (0c:ee:e6:94:8f:e5)

Internet Protocol Version 4, Src: 212.227.170.107, Dst: 172.31.222.216

Framsmission Control Protocol, Src Port: 80 (80), Dst Port: 41874 (41874), Seq: 60915, Ack: 341, Len: 629

[46 Reassembled TCP Segments (61543 bytes): #6527(1374), #6529(1374), #6531(1374), #6535(1374), #6537(1374), #6539(1374), #6541(1374)

[57] Hypertext Transfer Protocol

Compuserve GIF, Version: GIF89a
```

Figura 3: Esquema de encapsulamento do protocolo HTTP, contendo as camadas de: aplicação, transporte, rede, enlace e física

```
Wireshark Packet 7044 - pacotes_projeto1

Frame 7044: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0

Ethernet II, Src: Palladiu_4c:76:bf (5c:c9:d3:4c:76:bf), Dst: Broadcast (ff:ff:ff:ff)

Internet Protocol Version 4, Src: 172.31.192.13, Dst: 172.31.192.127

User Datagram Protocol, Src Port: 2008 (2008), Dst Port: 2008 (2008)

Data (20 bytes)
```

Figura 4: Esquema de encapsulamento do protocolo UDP, contendo as camadas de:

Figura 5: Esquema de encapsulamento do protocolo TCP, contendo as camadas de: transporte, rede, enlace e fisica

4 Formato do quadro IP, especificando e detalhando cada um dos conteúdos do cabeçalho.

• Version:

- Valor: 0100 = 4;
- 4 bits que especificam a versão do protocolo IP. A partir do número da versão, o roteador pode examinar o datagram IP. Foi capturado o datagrama com *Internet Protocol Version* 4 (IPv4).

• Header Length:

- Valor: 0101 = 5 * 32 bits = 160 bits = 20 bytes.
- Como um datagrama IPv4 pode conter número varável de opções incluidas no cabeçalho, estes 4 bits, que é o número de palavras de 32 bits, utilizados para determinar onde os dados começam. Foi capturado um datagrama com o valor do campo igual a 5 que indica o tamnho de 20 bytes.

```
Wireshark · Packet 7047 · pacotes_projeto1
Frame 7047: 695 bytes on wire (5560 bits), 695 bytes captured (5560 bits) on interface 0
Ethernet II, Src: ArubaNet_02:38:10 (00:1a:1e:02:38:10), Dst: HonHalPr_94:8f:e5 (0c:ee:e6:94:8f:e5)
Internet Protocol Version 4, Src: 212.227.170.107, Dst: 172.31.222.216
0100 ... = Version: 4
... 0101 = Header Length: 20 bytes

* Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
0000 00. = Differentiated Services Codepoint: Default (0)
... ... 00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 861
      Fragment offset: 0
Fragment offset: 0
Time to live: 44
Protocol: TCP (6)
"---der checksum: 0x6046 [validation disabled]
```

• Differentiated Services Length:

- Valor: 0x00, DSCP: Default, ECN: Not ECN-Capable Transport;
- Os bits de tipo de serviço foram incluídos no IPv4 para diferenciar o tipo de datagrama IP, por exemplo, que requerem baixo atraso, alta vazão ou confiabilidade. Pode ser útil para distinguir datagramas de tempo real como usados em VoIP.

• Total Length:

- Valor: 681;
- Comprimento total do datagrama IP (Cabeçalho e dados) medidos em bytes.

• Identification:

- Valor: 0xe1c1, Reserved Bit: Not set, Don't fragment: Set, More fragments: Not set;
- Identificação de um fragmento do datagrama.

• Flags:

- Valor: 0x02;
- Controle da fragmentação do datagrama.

• Fragment Offset:

- Valor: 0;
- Valor de deslocamento do fragmento.

• Time To Live:

- Valor: 44;
- Campo de tempo de vida de um datagrama, serve para garantir que o datagrama não fique circulando para sempre na rede. Esse campo é decrementado em uma unidade cada vez que o datagrama é processado por um roteador. Se TTL chegar a 0, o datagrama deve ser descartado.

• Protocol:

- Valor: 6 (TCP);
- O valor do campo indica o protocolo da camada de transporte específico ao qual a porção de dados desse datagrama IP deverá ser passada. O número de protocolo é o elo entre as camadas de rede e transporte.

• Header checksum:

- Valor: 0x6046;
- Serve para identificar datagramas recebidos com erros. É calculado tratando cada 2 bytes do cabeçalho como se fossem um número e somando estes números usando complementos aritméticos de 1. Um roteador calcula o valor do *checksum* para cada datagrama IP recebido, se um erro for detectado, em geral, o datagrama é descartado.

• Source:

- Valor: 212.227.170.107;IP de origem do pacote.
- Destination:
 - Valor: 172.31.222.216;IP de destino do pacote.
- Source GeoIP:
 - Valor: Germany, AS8510 1&1 Internet AG, 51, 9;
 - Informação geográfica do IP de origem do pacote.

• Destination GeoIP:

- Valor: Unknown;
- Informação geográfica do IP de destino do pacote.
- 5 O formato do cabeçalho do pacote Ethernet, na camada de enlace, descrevendo cada um dos campos e detalhando como funciona o controle de erro nesta camada a partir do conteúdo do campo CRC.

Figura 7: Formato do cabeçalho do pacote Ethernet.

• Destination:

- Valor: HonHaipR_94:8f:e5;
- Endereço MAC de destino.

• Source:

- Valor: ArubaNet₋02:38:10;
- Endereço MAC de origem.

• Type:

- Valor: 0x0800 = IPv4;
- Protocolo utilizado na camada de rede.

• Data:

- Valor:
- Dados.

• FCS:

- Valor:
- Campo de verificação da corretude da mensagem.

5.1 Cálculo do campo CRC.

O cálculo do campo CRC do pacote examinado é feito por meio de um polinômio gerador pré-definido g(x) = 0x04c11db7, chamado também de CRC-32. O conteúdo do pacote é deslocado de 26 bits para a esquerda (equivalente ao grau do polinômio gerador) gerando o encoding e(x). A partir disto, desloca-se também o polinômio gerador g(x) até o este ter o mesmo grau de e(x), assim são calculados sucessivas operações \mathbf{xor} , deslocando o g(x) sempre para o maior grau do resultado obtido. Ao final têm-se 4 bytes que definem o campo CRC do pacote.

Um programa na linguagem C (CRC32.c) foi desenvolvido para o cálculo do CRC do pacote específico 6904 (pacotes_projeto1.pcapng) que utiliza o protocolo Ethernet II na camada de enlace. O CRC de um pacote que utiliza Ethernet II é feito por meio de um código polinomial, que possui como polinômio gerador o CRC32, como já dito anteriormente. Mais detalhes sobre o cálculo do CRC deste pacote estão disponíveis no executável e na implementação em si.